GRAPH COLORING PROBLEMS AND THEIR APPLICATIONS IN SCHEDULING

DÁNIEL MARX

Department of Computer Science and Information Theory
Budapest University of Technology and Economics
H-1521 Budapest, Hungary
dmarx@cs.bme.hu

Abstract

Graph coloring and its generalizations are useful tools in modeling a wide variety of scheduling and assignment problems. In this paper we review several variants of graph coloring, such as precoloring extension, list coloring, multicoloring, minimum sum coloring, and discuss their applications in scheduling.

I. Introduction

A k*-coloring* of graph G is an assignment of integers $\{1, 2, \ldots, k\}$ (the colors) to the vertices of G in such a way that neighbors receive different integers. The *chromatic number* of G is the smallest k such that G has a k-coloring.

There are several interesting practical problems that can be modeled by graph coloring. Our basic example is the following. Assume that we have to schedule a set of interfering jobs, it has to be determined when each job is executed. Let G be the *conflict graph* of the jobs: the vertices of the graph corresponds to the jobs, two vertices are connected by an edge if the corresponding two jobs cannot be executed at the same time (for example, they use a shared resource or interfere in some other way). The colors correspond to the available time slots, every job requires one time slot. There is a one-to-one correspondence between the feasible schedulings of the jobs and the colorings of the graph: vertex v receives color i if and only if the corresponding job is executed in time slot i. Clearly, the graph has a k-coloring if and only if the jobs can be done in k time slots such that interfering jobs are not executed simultaneously. Therefore the chromatic number of the graph equals the minimum *makespan* of the scheduling problem, the minimum time required to finish the jobs.

Unfortunately, determining the chromatic number of a graph is an **NP**-hard problem, hence we cannot expect to solve it efficiently for large graphs. Therefore when we model a scheduling or assignment problem with graph coloring, then there are two things to consider. First, it might happen that the graphs arising in our application have special structure that makes coloring easier. Moreover, if there is no hope for an efficient (polynomial time) algorithm to the coloring problem that always finds an optimum solution, then we have to content ourselves with an *approximation algorithm* that is not optimal, but has some performance guarantee on the quality of the produced solution. For a minimization problem, we say that an algorithm is a c-approximation algorithm if it always produces a solution whose cost is at most c times the optimum.

We conclude this section by citing some specific applications of graph coloring where the conflict graph has special structure, and efficient optimal or approximation algorithm can be given for the problem.

Aircraft scheduling. Assume that we have k aircrafts, and we have to assign them to n flights, where the ith flight is during the time interval (a_i, b_i). Clearly, if two flights overlap, then we cannot assign the same aircraft to both flights. The vertices of the conflict graph correspond to the flights, two vertices are connected if the corresponding time intervals overlap. Therefore the conflict graph is an *interval graph*, which can be colored optimally in polynomial time.

Biprocessor tasks. Assume that we have a set of processors (machines) and a set of tasks, each task has to be executed on two preassigned processors simultaneously. A processor cannot work on two jobs at the same time. For example, such biprocessor tasks arise when we want to schedule file transfers between processors [1] or in the case of mutual diagnostic testing of processors [2].

Consider the graph whose vertices correspond to the processors, and if there is a task that has to be executed on processors i and j, then we add an edge between the two corresponding vertices. Now the scheduling problem can be modeled as an *edge coloring* of this graph: we have to assign colors to the edges in such a way that every color appears at most once at a vertex.

Edge coloring is **NP**-hard [3], but there are good approximation algorithms. The maximum degree Δ of the graph is an obvious lower bound on the number of colors needed to color the edges of the graph. On the other hand, if there are no multiple edges in the graph (there are no two tasks that require the same two processors), then Vizing's Theorem gives an efficient method for obtaining a $(\Delta + 1)$-edge coloring. If multiple edges are allowed, then the algorithm of [4] gives an 1.1-approximate solution.

Frequency assignment. Assume that we have a number of radio stations, identified by x- and y-coordinates in the plane. We have to assign a frequency to each station, but due to interferences, stations that are "close" to each other have to receive different frequencies. Such problems arise in frequency assignment of base stations in cellular phone networks.

At first sight, one might think that the conflict graph is planar in this problem, and the Four Color Theorem can be used, but it is not true: if there are lots of stations in small region, then they are all close to each other, therefore they form a large clique in the conflict graph. Instead, the conflict graph is a *unit disk graph*, where each vertex corresponds to a disk in the plane with unit diameter, and two vertices are connected if and only if the corresponding disks intersect. A 3-approximation algorithm for coloring unit disk graphs is given in [5], yielding a 3-approximation for the frequency assignment problem.

II. Multicoloring

A natural generalization of the basic setup introduced in Section I. is to consider jobs that require more than one time slots. In the *multicoloring* problem each vertex v has a *demand* $x(v)$, and we have to assign a set of $x(v)$ colors to each vertex v such that neighbors receive disjoint sets of colors. Multicoloring can be used to model the scheduling of jobs with different time requirements: the set of colors assigned to vertex v corresponds to the $x(v)$ time slots when we work on the job.

There are two main variants of multicoloring. In *non-preemptive* multicoloring the set of colors assigned to a vertex has to be a continuous interval of colors. This reflects the requirement that the jobs cannot be interrupted, they have to receive a continuous time window. On the other hand, in *preemptive* multicoloring we assume that the jobs can be interrupted, hence the set of colors assigned to a vertex can be arbitrary, it does not have to be continuous.

In [6] the frequency assignment problem discussed in Section I. is generalized, we have to assign a predefined number of frequencies to each base station. The authors adopt the hexagon model, which means that the conflict graph is a subgraph of the triangular lattice. A $\frac{4}{3}$-approximation algorithm is presented in [6] for minimizing the number of different frequencies assigned.

III. Precoloring extension

In certain scheduling problems we do not have full control over the schedule, the assignments of certain jobs are already decided. In this case some of the vertices of the conflict graph has a preassigned color, and we have to solve the *precoloring extension* problem: extended the coloring of these vertices to the whole graph, using the minimum number of colors.

Biró, Hujter and Tuza [7, 8, 9] started a systematic study of precoloring extension. In [7], the aircraft scheduling problem discussed in Section I. is extended. There is a maintenance period for each aircraft, during which it cannot fly. We can model these maintenance periods by adding a “dummy” flight for the maintenance period of each aircraft, and requiring that the maintenance period of the ith aircraft is assigned to the ith aircraft. Therefore we have to solve the precoloring extension problem on the conflict graph, which is an interval graph. It is shown in [7] that the precoloring extension problem is **NP**-complete for interval graphs, but it can be solved in polynomial time if every color is used only once in the precoloring, that is, if every aircraft has only one maintenance interval (the later result is generalized to chordal graphs in [10]).

IV. List coloring

In the list coloring problem each vertex v has a list of available colors, and we have to find a coloring where the color of each vertex is taken from its list of available colors. List coloring can be used to model situations where a job can be processed only in certain time slots, or if it can be processed only by certain machines.

Using standard dynamic programming techniques, list coloring can be solved in polynomial time on trees and partial k-trees [11]. By combining dynamic programming with a clever use matching, list coloring can be solved on the edges of trees as well [12].

The multicoloring concept introduced in Section II. can be applied for list colorings as well: each vertex has an integer demand $x(v)$, and vertex v has to receive a set of $x(v)$ colors from its list of colors. The algorithm for list coloring trees and partial k-trees does not generalize for the multicoloring case, as the problem is **NP**-complete already for binary trees [13]. On the other hand, list *edge* multicoloring can be solved in polynomial time on trees: using standard techniques, the good characterization theorem of Marcotte and Seymour [14] can be turned into a polynomial time algorithm. This result is generalized in [15] to a slightly more general class of graphs, that includes odd cycles. Moreover, a randomized algorithm is given for an even more general class of graphs, including even cycles.

V. Minimum sum coloring

Besides minimizing the makespan, another well-studied goal in scheduling theory is to minimize the sum of completion times of the jobs, which is the same as minimizing the average completion time. The corresponding coloring problem is *minimum sum coloring*, introduced in [16]: we are looking for a coloring of the conflict graph such that the sum of the colors assigned to the vertices is minimal.

Apart from trees, partial k-trees, and edges of trees, minimum sum coloring is **NP**-hard on most classes of graphs. On the other hand, it turns out that the sum of the coloring is easier to approximate than the makespan (see e.g. [17, 18] for approximation results). The reason for this is that the sum of the coloring and the makespan of the coloring behaves very differently when a small part of the graph is recolored. If we recolor a small part of the graph, then this change has only a small effect on the sum of the coloring, but it can change the makespan significantly.

The multicoloring version of the problem can be used to model arbitrary length jobs. Since we want to minimize the sum of the completion times, the objective function of the coloring problem has to be defined as follows. The *finish time* of a vertex is the largest color assigned to it, and the sum of a coloring is the sum of the finish times of the vertices. It is clear that the sum of the finish times in a multicoloring is equal to the sum of completion times in the corresponding schedule. This variant of multicoloring was introduced in [19], where approximation algorithms are given for various classes of graphs. The preemptive and non-preemptive versions of the problem can have very different complexity: while the non-preemptive version can be solved in polynomial time for trees [20], the preemptive version is **NP**-hard for binary trees [13], but has a polynomial time approximation scheme [20]. In [21] polynomial time approximation schemes are given for partial k-trees and planar graphs as well. Unlike minimum sum coloring, the multicoloring version of the problem is **NP**-hard on the edges of trees. However, in this case the problem admits a polynomial time approximation scheme [22].

VI. Acknowledgements

Research was supported by OTKA grants 44733, 42559 and 42706.

References

[1] E. G. Coffman, Jr., M. R. Garey, D. S. Johnson, and A. S. LaPaugh, "Scheduling file transfers," *SIAM J. Comput.*, 14(3):744–780, 1985.

[2] J. A. Hoogeveen, S. L. van de Velde, and B. Veltman, "Complexity of scheduling multiprocessor tasks with prespecified processor allocations," *Discrete Appl. Math.*, 55(3):259–272, 1994.

[3] I. Holyer, "The NP-completeness of edge-coloring," *SIAM J. Comput.*, 10(4):718–720, Nov. 1981.

[4] T. Nishizeki and K. Kashiwagi, "On the 1.1 edge-coloring of multigraphs," *SIAM J. Discrete Math.*, 3(3):391–410, 1990.

[5] A. Gräf, M. Stumpf, and G. Weißenfels, "On coloring unit disk graphs," *Algorithmica*, 20(3):277–293, 1998.

[6] L. Narayanan and S. M. Shende, "Static frequency assignment in cellular networks," *Algorithmica*, 29(3):396–409, 2001.

[7] M. Biró, M. Hujter, and Zs. Tuza, "Precoloring extension. I. Interval graphs," *Discrete Math.*, 100(1-3):267–279, 1992.

[8] M. Hujter and Zs. Tuza, "Precoloring extension. II. Graph classes related to bipartite graphs," *Acta Mathematica Universitatis Comenianae*, 62(1):1–11, 1993.

[9] M. Hujter and Zs. Tuza, "Precoloring extension. III. Classes of perfect graphs," *Combin. Probab. Comput.*, 5(1):35–56, 1996.

[10] D. Marx, "Precoloring extension on chordal graphs," 2003, submitted.

[11] K. Jansen and P. Scheffler, "Generalized coloring for tree-like graphs," *Discrete Appl. Math.*, 75(2):135–155, 1997.

[12] K. Giaro and M. Kubale, "Edge-chromatic sum of trees and bounded cyclicity graphs," *Inform. Process. Lett.*, 75(1-2):65–69, 2000.

[13] D. Marx, "The complexity of tree multicolorings," in *Mathematical Foundations of Computer Science 2002 (Warsaw-Otwock)*, pp. 532–542. Springer, Berlin, 2002.

[14] O. Marcotte and P. D. Seymour, "Extending an edge-coloring," *J. Graph Theory*, 14(5):565–573, 1990.

[15] D. Marx, "List edge multicoloring in graphs with few cycles," *Inform. Process. Lett.*, 89(2):85–90, 2004.

[16] E. Kubicka and A. J. Schwenk, "An introduction to chromatic sums," in *Proceedings of the ACM Computer Science Conf.*, pp. 15–21. Springer, Berlin, 1989.

[17] A. Bar-Noy, M. Bellare, M. M. Halldórsson, H. Shachnai, and T. Tamir, "On chromatic sums and distributed resource allocation," *Inform. and Comput.*, 140(2):183–202, 1998.

[18] K. Giaro, R. Janczewski, M. Kubale, and M. Małafiejski, "A $27/26$-approximation algorithm for the chromatic sum coloring of bipartite graphs," in *Proceedings of APPROX 2002*, pp. 135–145, 2002.

[19] A. Bar-Noy, M. M. Halldórsson, G. Kortsarz, R. Salman, and H. Shachnai, "Sum multicoloring of graphs," *J. Algorithms*, 37(2):422–450, 2000.

[20] M. M. Halldórsson, G. Kortsarz, A. Proskurowski, R. Salman, H. Shachnai, and J. A. Telle, "Multicoloring trees," *Inform. and Comput.*, 180(2):113–129, 2003.

[21] M. M. Halldórsson and G. Kortsarz, "Tools for multicoloring with applications to planar graphs and partial k-trees," *J. Algorithms*, 42(2):334–366, 2002.

[22] D. Marx, "Minimum sum multicoloring on the edges of trees," 2003, To appear in 1st Workshop on Approximation and Online Algorithms, Budapest, 2003.